Naval Research Laboratory

Washington, DC 20375-5000

NRL Memorandum Report 6797

A Simple Technique for Displaying WDB-II Digital Cartographic Imagery in KEE

JAMES A. KILGORE, JR.

Battle Management Technology Branch
Information Technology Division

May 7, 1991

REPORT DOCUMENTATION PAGE

Form Approved
OMB No 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	1991 May 7	

4. TITLE AND SUBTITLE

A Simple Technique for Displaying WDB-II Digital Cartographic Imagery in KEE

5. FUNDING NUMBERS

JO# 55-2579-A-1
PE# 62232N
TASK# #
WUAS# 89-349

6. AUTHOR(S)

James A. Kilgore, Jr.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

NRL Code 5570, Commanding Officer
4555 Overlook Ave., SW
Washington, DC 20375-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

NRL Memorandum Report 6797

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

ONT - Office of Naval Technology
800 N. Quincy St.
Arlington, VA 2217-5000

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

A technique for displaying World Data Bank II digital cartographic imagery in KEE using Keepictures is described. World Data Bank II is a cartographic database of the world distributed by the Central Intelligence Agency and divides the world into five geographic areas that are represented by unique line segments defined by individually digitized points. Keepictures is a computer graphics toolkit that is a part of the Knowledge Engineering Environment (KEE) which is a commercial software development product by Intellicorp, Inc.

14. SUBJECT TERMS

Cartographic Database, Kee, Digital Imagery, World Databank-II

15. NUMBER OF PAGES

34

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	SAR

Contents

A SIMPLE TECHNIQUE FOR DISPLAYING WDB-II DIGITAL CARTOGRAPHIC IMAGERY IN KEE

1 Introduction

The Knowledge Engineering Environment (KEE) is a sophisticated commercial software development environment by IntelliCorp, Inc., Mountain View, California. KEE contains a computer graphics toolkit called KEEpictures for building graphical software to be used with KEE. KEEpictures is an object-oriented computer graphics application that is fully integrated with the KEE system and provides the developer with a wide range of capability from rendering simple displays to implementing complicated animation techniques [1].

Software systems developed using KEE often have a requirement to display maps. It is not hard to integrate cartographic databases with KEE, and there is an abundance of commercial databases to choose from. The Central Intelligence Agency (CIA) and Defense Mapping Agency (DMA) are two excellent sources of digital map data within the defense community, for example. This paper will describe a simple technique for displaying maps drawn from the World Data Bank II cartographic database obtained from the CIA. The maps were displayed on a Sun workstation in KEE using KEEpictures.

2 Description of the Hardware and Software Environment

A Sun 3/260 workstation running Sun operating system (Sun OS) version 4.1 was used as the development platform for this work. KEE version 3.1.117 was used with Sun Common Lisp version 2.1.3, and this version of KEEpictures supported only monochrome graphical operations. The *draw* program described in section 6 was written using the SunCore computer graphics software library. SunCore was a software product by Sun Microsystems that had been bundled with the Sun OS prior to version 4.0. Sun has discontinued support for the SunCore product in Sun OS version 4.0 and later releases, and consequently SunCore libraries and include files are no longer distributed. These files were ported from the Sun OS 3.5 environment to the 4.1 environment without any noticeably deleterious side effects.

3 Approach

The technical approach is described briefly in the following five steps, and figure 1 shows the processing pipeline.

1. Read the geographic areas of interest from files on World Data Bank II (WDB-II) distribution tapes and convert the files from WDB-II file format to files in a simpler file format, called *sff* files.
2. Optionally filter or correct the data.
3. Draw the map on the Sun console.
4. Save a screendump of the image in a file.

Manuscript approved February 15, 1991

5. Convert the raster image into a lisp expression that, when loaded using the KEE lisp listener, creates a Common Windows bitmap of the image. Write this lisp expression to a file.
6. Load the file in KEE using the lisp listener, creating a Common Windows bitmap structure. The map can be viewed in a KEEpictures viewport by making the bitmap structure the value of the bitmap slot in a KEEpictures bitmap unit.

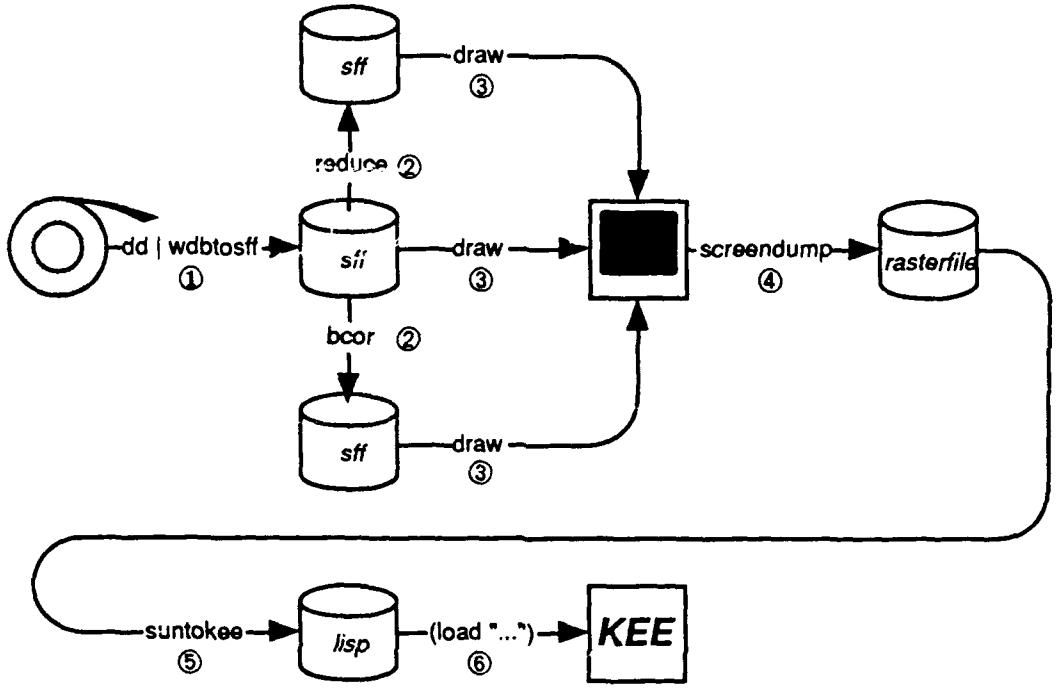

Figure 1.

4 World Data Bank II

World Data Bank II is a cartographic database of the world distributed by the Central Intelligence Agency (CIA), Washington, DC. WDB-II is a digital representation of the world divided into five geographic areas. Each area is contained on a single magnetic tape volume and each tape contains three files of unique line segments defined by individually digitized points. The five geographic areas are North America, South America, Europe, Africa, and Asia, and the three files contained in each tape volume are a) coastlines, islands, and lakes, b) rivers, and c) international boundaries. In addition, the North America volume contains a fourth file of internal boundaries to further delineate the individual states of the USA and the provinces of Canada. WDB-II was digitized at scales ranging from 1:1 million to 1:4 million; the nominal scale is 1:3 million. WDB-II contains approximately 6 million digitized points. [2]

4.1 Description of WDB-II Files

Each of the 16 WDB-II files consist of a sequence of line segments and are formatted on the tape in the following way. Each line segment begins with a 20 character record:

Columns	1 - 7	Line segment number
	8 - 9	Rank
	10 - 15	Number of points in line
	16 - 20	Zeroes

Each point in the line segment follows and they are described by this 20 character record:

Columns	1 - 7	Latitude (DDMMSSI)
	8 - 15	Longitude (DDMMSSI)
	16 - 20	Point number

Each line segment in WDB-II has a numeric rank assigned to it which is described below:

Coast, islands, and lakes

01	Coasts, islands, and lakes that appear on all maps
02	Additional major islands and lakes
03	Intermediate islands and lakes
04	Minor islands and lakes
06	Intermittent major lakes
07	Intermittent minor lakes
08	Reefs
09	Major salt pans
10	Minor salt pans
13	Major ice shelves
14	Minor ice shelves
15	Glaciers

Rivers

01	Permanent major rivers
02	Additional major rivers
03	Additional rivers
04	Minor rivers
05	Double-lined rivers
06	Major intermittent rivers
07	Additional intermittent rivers
08	Minor intermittent rivers
10	Major canals
11	Canals of lesser importance
12	Irrigation type canals

International boundaries or limits of sovereignty

01	Demarcated or delimited
02	Indefinite or in dispute
03	Other line of separation or sovereignty on land

Internal boundaries

01	First order administrative

4.2 Reading WDB-II Files from Tape and Converting them to sff Files

The WDB-II data files tape format specification was not very convenient for our purposes so a special program was used, called *wdbtosff*, to convert the data into a form that is easier to use. The *wdbtosff* program reads a WDB-II file and converts it into an ASCII text file called an sff file (which stands for simpler file format) given by the following grammar:

<sff file>	→	<segment list> EOF
<segment list>	→	[<segment list>]
	→	<rank> <separator> <n points> <separator> <point list>
<point list>	→	<point> [<point list>]
<point>	→	<lat> <separator> <lon> <separator>
<rank>	→	<character integer>
<n points>	→	<character integer>
<separator>	→	Space | CarriageReturn [<separator>]
<character integer>	→	[+ | -] <digits>
<digits>	→	<digit> [<digits>]
<digit>	→	0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<lat>	→	<character float>
<lon>	→	<character float>
<character float>	→	<character integer> [. <digits>]

The value of <n points> describes the number of instances of <point> in <point list> which follow.

WDB-II files can be read directly from the distribution tape using the UNIX *dd* command and piped to the *wdbtosff* program, whose output is redirected to a file. The following examples show this in detail for each WDB-II file. Note that files on the Asia volume have a block size of 8000 bytes; all other files have a block size of 3600 bytes. Note also that WDB-II tapes are written in the EBCDIC character set and must be converted to ASCII using the conv=ascii option.

Vol 1: North America

```
dd if=/dev/nrmt0 bs=3600 conv=ascii | wdbtosff > na.cil
dd if=/dev/nrmt0 bs=3600 conv=ascii | wdbtosff > na.riv
dd if=/dev/nrmt0 bs=3600 conv=ascii | wdbtosff > na.bdy
dd if=/dev/nrmt0 bs=3600 conv=ascii | wdbtosff > na.pby
```

Vol 2: South America

```
dd if=/dev/nrmt0 bs=3600 conv=ascii | wdbtosff > sa.cil
dd if=/dev/nrmt0 bs=3600 conv=ascii | wdbtosff > sa.riv
dd if=/dev/nrmt0 bs=3600 conv=ascii | wdbtosff > sa.bdy
```

Vol 3: Europe

```
dd if=/dev/nrmt0 bs=3600 conv=ascii | wdbtosff > eur.cil
dd if=/dev/nrmt0 bs=3600 conv=ascii | wdbtosff > eur.riv
dd if=/dev/nrmt0 bs=3600 conv=ascii | wdbtosff > eur.bdy
```

Vol 4: Africa

```
dd if=/dev/nrmt0 bs=3600 conv=ascii | wdbtosff > af.cil
dd if=/dev/nrmt0 bs=3600 conv=ascii | wdbtosff > af.riv
```

```
dd if=/dev/nrmt0 bs=3600 conv=ascii | wdbtosff > af.bdy
```

Vol 5: Asia

```
dd if=/dev/nrmt0 bs=8000 conv=ascii | wdbtosff > as.cil
dd if=/dev/nrmt0 bs=8000 conv=ascii | wdbtosff > as.riv
dd if=/dev/nrmt0 bs=8000 conv=ascii | wdbtosff > as.bdy
```

The source code of the *wdbtosff* program is shown in Appendix A.

5 Optionally Filtering the Data or Correcting Boundary Errors

A data reduction program called *reduce* is shown in Appendix B and can be used to filter data points against three criteria: minimum and maximum distance between two points and angle described by three points. Roughly, an adjacent point is discarded if the distance between the test point and adjacent point does not lie within a range specified by the user, or, if the angle described by three sequential points is near 180 degrees then the middle point is discarded. A precise description of the algorithm can be found in the source code listing in Appendix B.

WDB-II data point values are bounded by [90S, 90N] in latitude and [180W, 180E] in longitude. The files as.riv, as.cil and sa.cil contain some line segments that cross the international date line, which may cause problems for unsophisticated drawing programs (such as the one described next). If a point that crosses the date line is not accounted for correctly a line connecting a pair of points on either side of the date line will be drawn erroneously across the map. A simple program called *bcor* corrects this problem by splitting the offending segment into new segments, each of which reaches the date line but does not cross it. The source code listing for *bcor* is given in Appendix C.

6 Drawing the Map and Making a Screendump of the Image

Maps are drawn on the Sun console with a program called *draw*, which projects the data points using a quasi-equirectangular projection. Here is the usage:

```
draw m|M|c rankmax latmin lonmin latmax lonmax file1 [file2 file3 ...]
```

Draw takes several arguments, the first of which is a single character that identifies the type of display console to the program. The three choices are *m* which stands for low resolution (1152x900 pixel) monochrome, *M* for high resolution (1600x1280 pixel) monochrome, or *c* for low resolution (1152x900 pixel) color. Segments with rank value greater than *rankmax* are not drawn. The next four arguments define two points on the globe which describe an area that is projected onto the display console: *latmin*, *lonmin*, *latmax*, and *lonmax*. Latitudes south of the equator and longitudes west of the prime meridian are represented as negative floating point numbers; all others are represented as positive floating point numbers. The arguments that follow are names of sff files that are to be drawn on the console. For example, to draw the whole world on a high-resolution monochrome console the following command line would be given:

```
draw M 15 -90 -180 90 180 na.cil na.riv na.bdy na.pby sa.cil sa.riv sa.bdy eur.cil
     eur.riv eur.bdy af.cil af riv af.bdy as.cil as.riv as.bdy
```

To draw European coastlines, islands, lakes, rivers, and international boundaries of rank 1-3 projected from 25N 30W to 75N 35E on a low resolution color console the following command line would be given:

draw c 3 25 -30 75 35 eur.cil eur.riv eur.bdy

When the drawing is finished the image should be saved in a file using the screendump command:

screendump > theworld.ras

Note: the *draw* program should not be run from the console because feedback from *draw* will appear on the screen and ruin the image. The source code listing for *draw* is shown in Appendix D.

Figures 2-5 show several example drawings. Figure 2 shows all WDB-II data. The *draw* command used here was:

draw M 15 -90 -180 90 180 box2 box3 af.bdy af.cil af.riv as.bdy as.cil as.riv
eur.bdy eur.cil eur.riv na.bdy na.cil na.pby na.riv sa.bdy sa.cil sa.riv

Box2 and box3 are small files that contain data for drawing rectangles. Box2 was used to draw the rectangle enclosing the whole map. Note the small rectangle around the Persian Gulf region. Box3 was used to draw this rectangle. Figure 3 shows an enlarged view of this region. The contents of box2 and box3 are shown in Appendix F.

Figure 3 shows the Persian Gulf region bounded by 23N 42E and 32N 60E. The *draw* command used to make this picture was:

draw M 15 23 42 32 60 box3 box4 af.bdy af.cil af.riv as.bdy as.cil as.riv

Box3 was used to draw the rectangle enclosing the map, and box4 was used to draw the small rectangle in the Kuwait region. Figure 4 shows an enlarged view of this region. The contents of box3 and box4 are shown in Appendix F.

Figure 4 shows the Persian Gulf region bounded by 23N 42E and 32N 60E. The *draw* command used to make this picture was:

draw M 15 23 42 32 60 box4 af.bdy af.cil af.riv as.bdy as.cil as.riv

Box4 was used to draw the rectangle enclosing the map. The contents of are shown in Appendix F.

Figure 5 shows a less cluttered view of coastlines, islands, lakes, and international boundaries of the world, and internal boundaries of North America. The *draw* command used to make this picture was:

draw M 1 -90 -180 90 180 box2 af.bdy af.cil as.bdy as.cil eur.bdy eur.cil na.bdy na.cil na.pby sa.bdy sa.cil

Box2 was used to draw the rectangle around the map. The contents of box2 are shown in Appendix F.

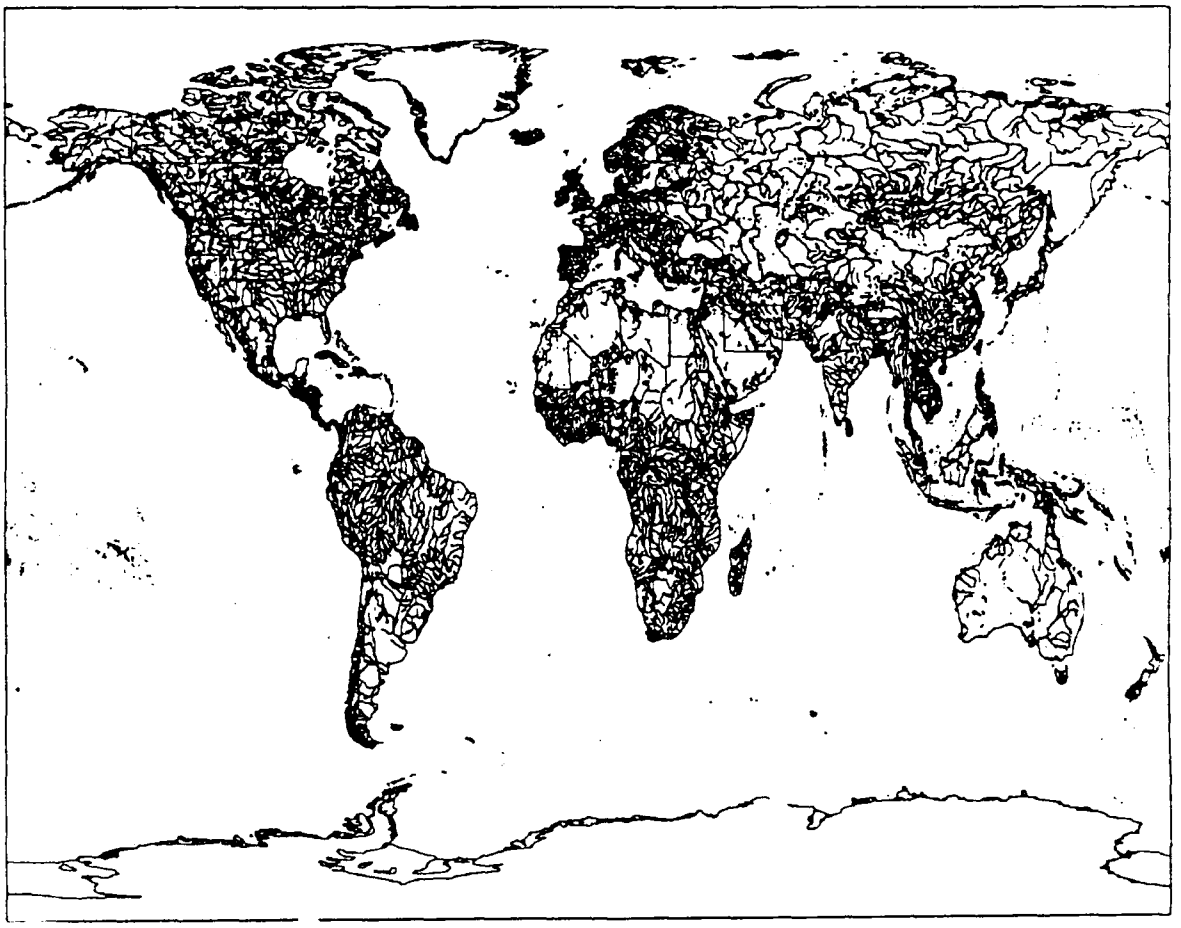

Figure 2.

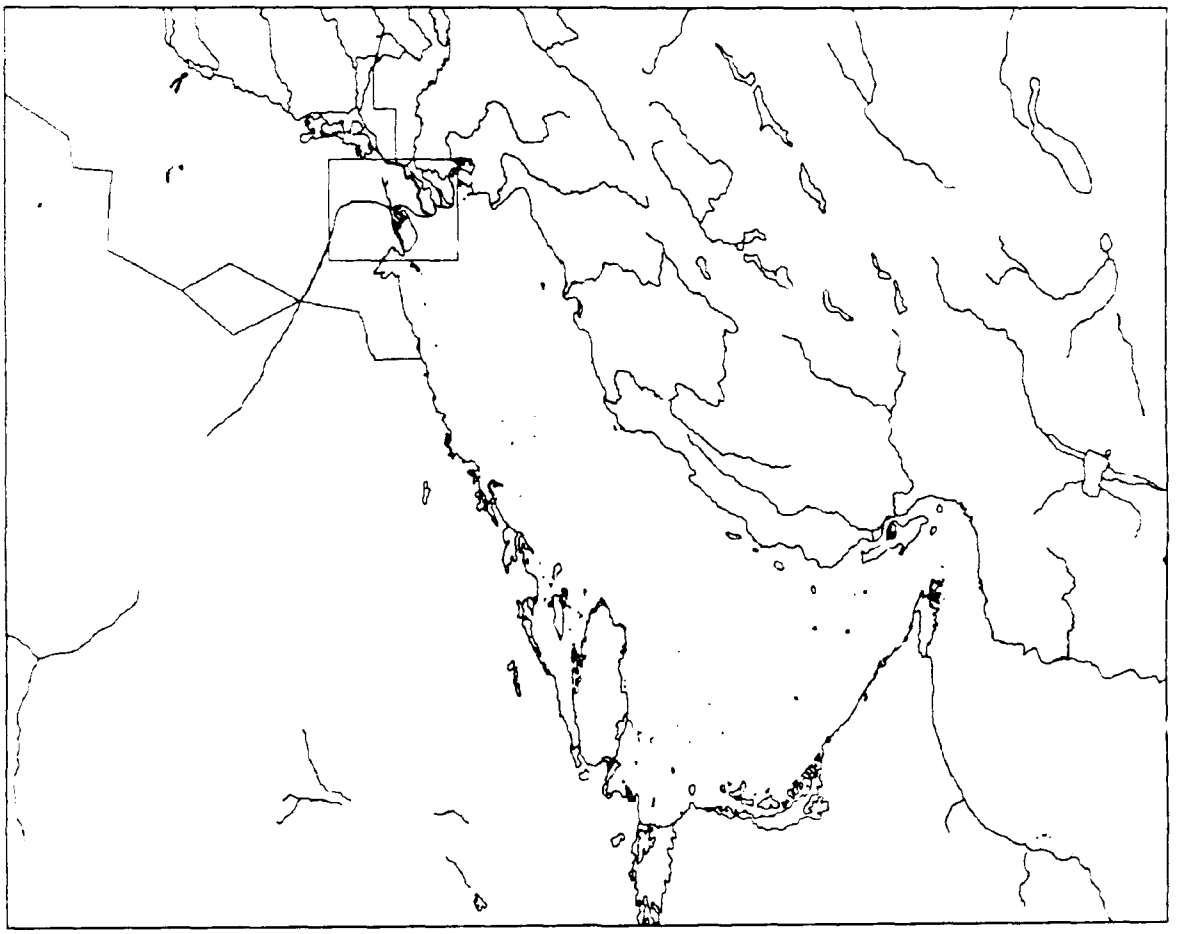

Figure 3.

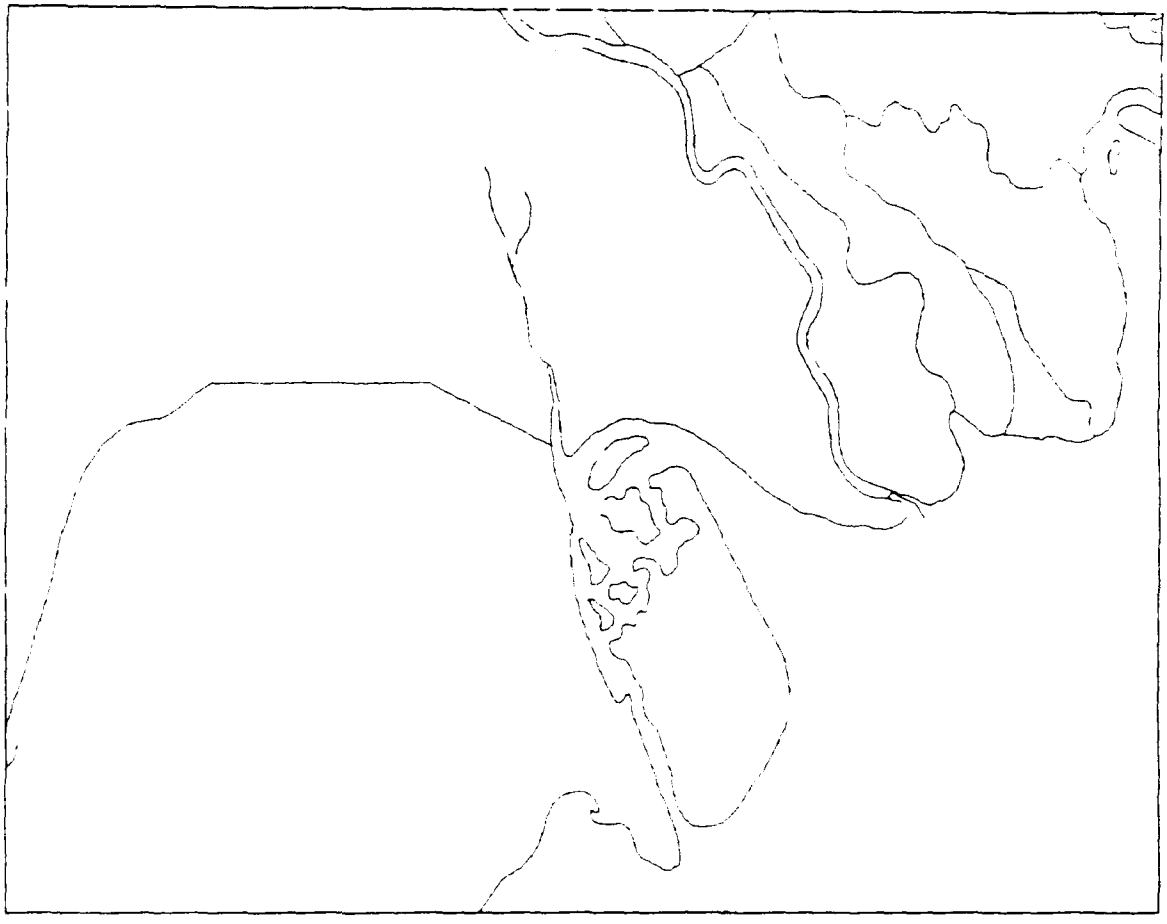

Figure 4.

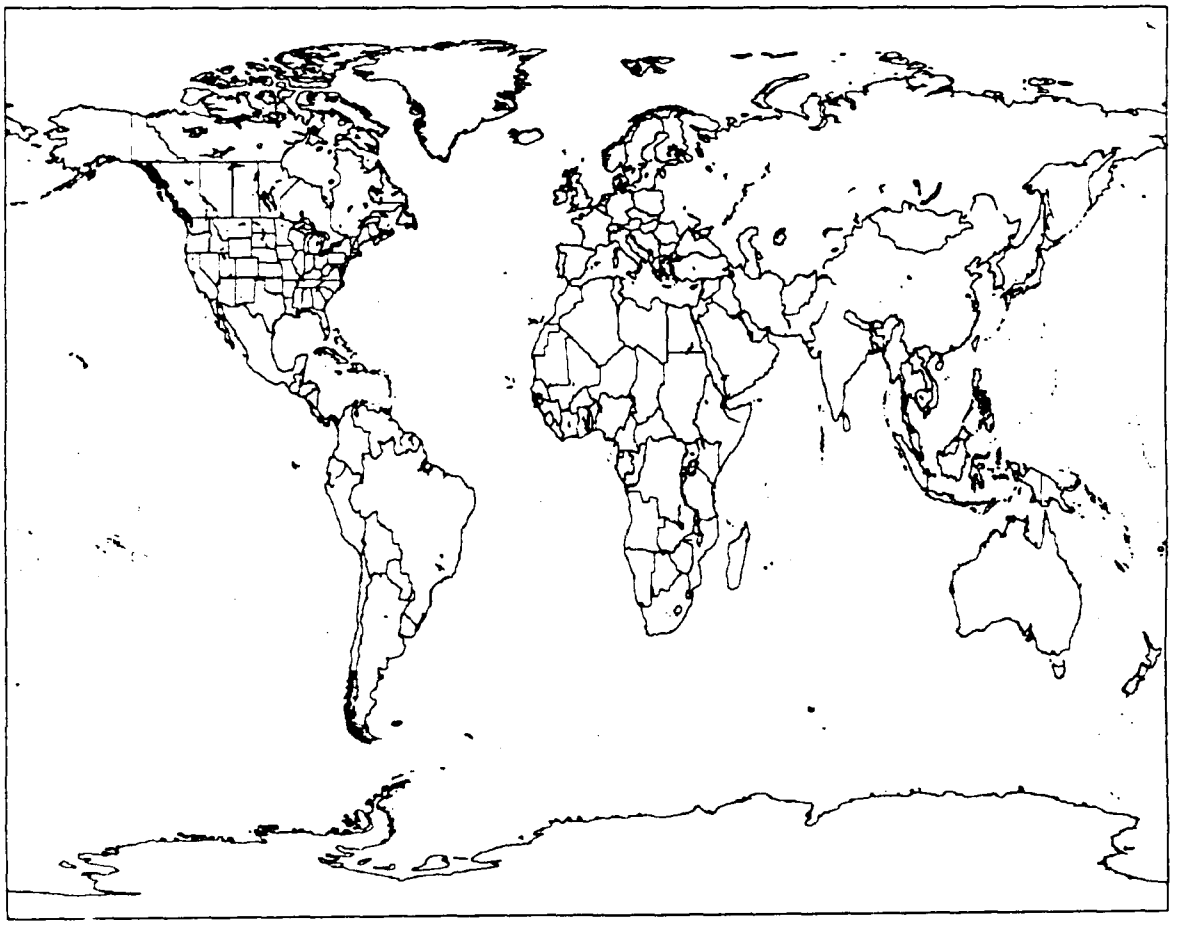

Figure 5.

7 Making a Common Windows Bitmap File

A program called *suntokee* converts a raster image saved as a monochrome rasterfile into a file that contains a lisp expression which, when loaded by the KEE lisp listener, assigns a Common Windows bitmap structure containing the identical raster image to a lisp variable. The variable name *bitmappname* is specified as an argument to the suntokee program:

suntokee bitmappname < rasterfile > lispfile

To define a Common Windows bitmap named kuwait-bitmap in the file called kuwait-bitmap.lisp from the screendump image in the 1600x1280 rasterfile kuwait.ras, for example, the following line would be used:

suntokee kuwait-bitmap < kuwait.ras > kuwait-bitmap.lisp

The lisp source file is a simple ASCII text file and can be viewed using an editor or printed out by the curious user. Also, the file can be compiled using the lisp function *compile-file* for greater efficiency.

The *suntokee* program is shown in Appendix E.

8 Viewing the Map in KEE

The next step is to load the lisp file with the KEE lisp listener, using the lisp *load* function. Having created a file called kuwait-bitmap.lisp using the example in section 7, the following line would be entered in the KEE lisp listener:

(load "kuwait-bitmap.lisp")

This creates a 1600x1280 Common Windows bitmap structure named kuwait-bitmap. To view the bitmap first make a KEEpictures viewport programmatically with (unitmsg 'viewports 'create.instance! ...), or interactively using the mouse and menu. Next create a KEEpictures bitmap unit in the viewport. The bitmap unit contains a slot called *bitmap*, which always contains a reference to a Common Windows bitmap structure. Set the value of the bitmap slot of this KEEpictures bitmap unit to kuwait-bitmap programmatically with (put.value ...), or interactively using the mouse and menu. Figure 6 shows the raster image from the rasterfile in the KEEpictures viewport.

Since the geometry of the cartographic projection and the geographic boundaries of the bitmap image are known it is a straightforward job to implement coordinate system transforms from geographic coordinate space to KEEpicture coordinate space and back.

9 References

[1] KEE User's Manual, IntelliCorp, 1989.

[2] World Data Bank II User Guide, Central Intelligence Agency, July, 1977.

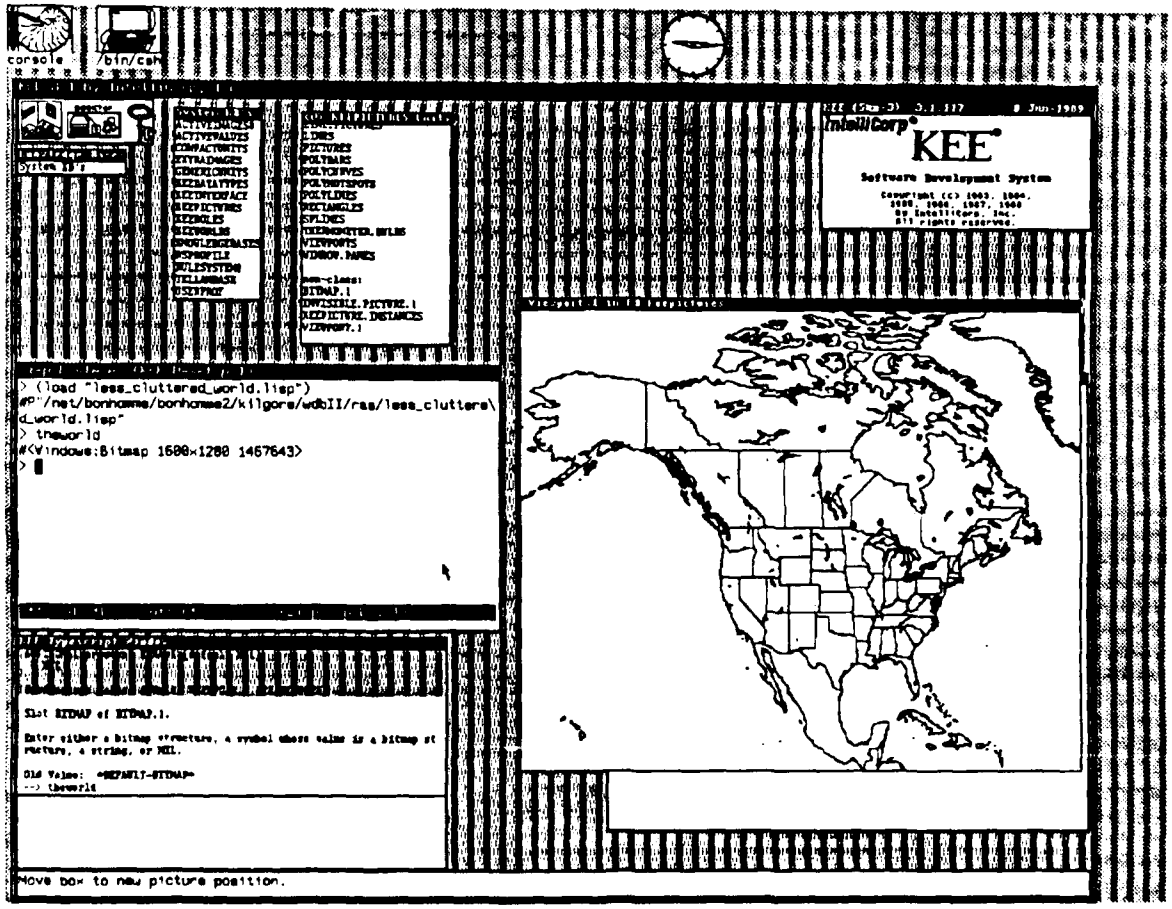
KEE
Software Development System
> (load "less_cluttered_world.lisp")
> theworld
Move box to new picture position.

Figure 6.

Appendix A

wdbtosff Program Source Code Listing

```
/* WDBTOSFF:  a simplistic program that reads a file in WDB-II format
   from stdin and writes ASCII text in the following format to stdout:

        <segment1>
        <segment2>
        .
        .
        .

   where each segment is:

        <rank> <n>
        <lat1> <lon1>
        <lat2> <lon2>
        .
        .
        .
        <latn> <lonn>

   WDB-II data is typically read from tape.  Here is an example in
   which the unix command dd is used to read the tape:

   dd if=/dev/nrmt0 bs=3600 conv=ascii | wdbtosff > eur.cil

   Written by Jim Kilgore
   U. S. Naval Research Laboratory
   February 1989 */

#include <stdio.h>
#include <sys/file.h>

main(argc,argv)
int argc;
char **argv;
  {
  FILE *ofp=stdout,*fopen();
  register int linesegno,rank,ifd=fileno(stdin),npts,i;
  register float latdeg,latmin,latsec,longdeg,longmin,longsec,llatdeg,llongdeg;
  char *strncpy(),ch2[3],ch3[4],ns[2],ew[2],*pname=argv[0];
  char ch20[21],ch6[7],ch7[8],ch8[9];

  if (argc != 1)
    {
    (void) fprintf (stderr, "usage: %s: < infname > outfname\n", pname);
    return -1;;
    }

  ch20[20]='\0';
  ch6[6]='\0';
  ch2[2]='\0';
  ch7[7]='\0';
  ch8[8]='\0';
  ns[1]='\0';
  ew[1]='\0';

  while (1)
    {
    if (read(ifd,ch20,20) == 0)
```

```
    {
    (void) fprintf (stderr, "%s: EOF while reading 1st ch20\n", pname);
    return -1;
    }

  (void) strncpy (&ch7[0],&ch20[0],7);

  linesegno=atoi(ch7);

  if ((linesegno != 9) && (linesegno != 900000) && (linesegno != 999999))
    {
    (void) strncpy (&ch6[0],&ch20[9],6);
    npts=atoi(ch6);

    (void) strncpy (&ch2[0],&ch20[7],2);
    rank=atoi(ch2);

    (void) fprintf (ofp,"%d %d\n",rank,npts);
    (void) fflush (ofp);

    for (i=0;i<npts;i++)
      {
      if (read(ifd,ch20,20) == 0)
        {
        (void) fprintf (stderr, "%s: EOF while reading 2nd ch20\n", pname);
        return -1;
        }
      (void) strncpy (&ch2[0],&ch20[0],2);
      latdeg=(float)atoi(ch2);
      (void) strncpy (&ch2[0],&ch20[2],2);
      latmin=(float)atoi(ch2);
      (void) strncpy (&ch2[0],&ch20[4],2);
      latsec=(float)atoi(ch2);
      (void) strncpy (&ns[0],&ch20[6],1);
      llatdeg=latdeg+latmin/60.+latsec/3600.;
      if (ns[0] == 'S')
        {
        llatdeg= -llatdeg;
        }
      (void) strncpy (&ch3[0],&ch20[7],3);
      longdeg=(float)atoi(ch3);
      (void) strncpy (&ch2[0],&ch20[10],2);
      longmin=(float)atoi(ch2);
      (void) strncpy (&ch2[0],&ch20[12],2);
      longsec=(float)atoi(ch2);
      (void) strncpy (&ew[0],&ch20[14],1);
      llongdeg=longdeg+longmin/60.+longsec/3600.;
      if (ew[0] == 'W')
        {
        llongdeg= -llongdeg;
        }
      (void) fprintf (ofp,"%f %f\n",llatdeg,llongdeg);
      (void) fflush (ofp);
      }
    }
  else
    {
    (void) fprintf (stderr, "%s: just read terminating record\n", pname);
    return 0;
    }
  }
}
```

Appendix B

reduce Program Source Code Listing

```
/* REDUCE: a simple data reduction algorithm which discards points i
   based on three criteria:  minimum distance between two adjacent
   points, maximum distance between two adjacent points, and the
   angle described by three sequential points.

   For each line segment (p1, p2, p3, ...) three sequential points are
   considered: p1, p2, and p3.  The distances between p1-p2 and p2-p3
   are measured, and if either is <= a minimum distance threshold
   then point p2 is discarded.  Otherwise, if either distance is >= a
   maximum distance threshold then point p2 is discarded.  If p2
   hasn't been discarded yet then the angle A described by the three
   points is measured.  If A >= max angle threshold T, where
   0 <= T <= 180 degrees then p2 is discarded.  If p2 was discarded in
   any of the above 3 cases then p2 is now set to p3, and p3 is set to
   p4.  Otherwise, if p2 was retained then p1 is set to p2, p2 is set
   to p3, and p3 is set to p4.  The process begins again considering
   p1, p2, and p3, and continues in this manner until the end of the
   segment is reached for all segments.

   Written by Jim Kilgore
   Naval Research Laboratory */


#include <stdio.h>
#include <math.h>
#include <sys/file.h>

#define TRUE 1
#define FALSE 0
#define RADPERDEG 0.01745329
#define PI 3.1415
#define BUFLEN 30000

#define dist(x1, y1, x2, y2) (sqrt ((x2-x1)*(x2-x1)+(y2-y1)*(y2-y1)))

double retainedlat[BUFLEN],retainedlong[BUFLEN];

main(argc,argv)
int argc;
char **argv;
  {
  FILE *ifp=stdin,*ofp=stdout,*fopen();
  int totalpoints,totalretained,discardp2,eof,rank,npts,i,nretained;
  int mindistcrit,maxdistcrit,curvcrit;
  float p1lat,p1long,p2lat,p2long,p3lat,p3long;
  double maxthreshdeg,threshrad,atof(),theta123,sin(),cos(),sqrt(),acos();
  double minthreshdist,maxthreshdist;
  register double d,a,a2,b,b2,c,c2,x;
  char *pname=argv[0];

  if (argc != 4)
    {
    (void) fprintf (stderr, "usage: %s maxthreshdeg minthreshdist maxthreshdist
< ifname > ofname\n",pname);
    return -1;
    }
```

```
  maxthreshdeg=atof(argv[1]);

  if (maxthreshdeg < 0.0 || maxthreshdeg > 180.0)
    {
    (void) fprintf (stderr, "%s: 0.0 <= maxthreshdeg <= 180.0\n",pname);
    return -1;
    }

  threshrad=maxthreshdeg*RADPERDEG;

  minthreshdist=atof(argv[2]);

  if (minthreshdist < 0.0)
    {
    (void) fprintf (stderr, "%s: minthreshdist must be >= 0.0\n",pname);
    return -1;
    }

  maxthreshdist=atof(argv[3]);

  eof=FALSE;
  totalpoints=0;
  totalretained=0;

  while (!eof)
    {
    if (fscanf(ifp,"%d%d",&rank,&npts) == EOF)
      {
      eof=TRUE;
      }

    else
      {
      (void) fprintf (stderr, "%s: rank=%d, npts=%d\n",pname,rank,npts);

      totalpoints=totalpoints+npts;
      mindistcrit=0;
      maxdistcrit=0;
      curvcrit=0;
      nretained=0;

      if (npts == 1)
        {
        /* Read point but reject set: */

        if (fscanf(ifp,"%f%f",&p1lat,&p1long) == EOF)
          {
          (void) fprintf (stderr, "%s: unexpected EOF while reading p1\n",pname)
;
          return -1;
          }
        }

      else if (npts == 2)
        {
        /* Read points and test for mindistcrit, maxdistcrit: */

        if (fscanf(ifp,"%f%f%f%f",&p1lat,&p1long,&p2lat,&p2long) == EOF)
{
          (void) fprintf (stderr, "%s: unexpected EOF while reading p1, p2\n",pn
ame);
          return -1;
          }
```

```
        d=dist(p1long,p1lat,p2long,p2lat);

        if (d <= minthreshdist)
          mindistcrit=mindistcrit+2;

        else if (d >= maxthreshdist)
          maxdistcrit=maxdistcrit+2;

        else
          {
          (void) fprintf (ofp, "%d %d\n",rank,npts);
          (void) fprintf (ofp, "%f %f\n%f %f\n",p1lat,p1long,p2lat,p2long);
          nretained=2;
          }
        }

      else
        {
        if (fscanf(ifp,"%f%f%f%f",&p1lat,&p1long,&p2lat,&p2long) == EOF)
          {
          (void) fprintf (stderr, "%s: unexpected EOF while reading p1, p2\n",pn
ame);
          return -1;
          }

        /* First point is retained:  */

        retainedlat[0]=p1lat;
        retainedlong[0]=p1long;
        nretained++;

        for (i=0;i<npts-2;i++)
          {
          if (fscanf(ifp,"%f%f",&p3lat,&p3long) == EOF)
            {
            (void) fprintf (stderr, "%s: unexpected EOF while reading p3\n",pnam
e);
            return -1;
            }

          a=dist(p1long,p1lat,p3long,p3lat);
          b=dist(p1long,p1lat,p2long,p2lat);
          c=dist(p2long,p2lat,p3long,p3lat);

          if (b <= minthreshdist || c <= minthreshdist)
            {
            (void) fprintf (stderr, "%s: mindistcrit violated\n",pname);
            discardp2=TRUE;
            mindistcrit++;
            }

          else if (b >= maxthreshdist || c >= maxthreshdist)
            {
            (void) fprintf (stderr, "%s: maxdistcrit violated\n",pname);
            discardp2=TRUE;
            maxdistcrit++;
            }

          else
            {

            /* Law of cosines:  a^2 = b^2 + c^2 - 2bc cos A:        */

            a2=a*a;
```

```
            b2=b*b;
            c2=c*c;
            x= (b2+c2-a2)/(2.*b*c);

            /* Normalize for ocassional roundoff error in boundary cases:  */

            if (x < -1.0)     /* if x < -1.0 it is just SLIGHTLY           */
              x= -1.0;        /* < -1.0 and should be corrected to -1.0    */

            if (x > 1.0)
              x=1.0;

            theta123=acos(x);

            /* Normalize for ocassional roundoff error in boundary cases: */

            if (theta123 < 0.0)  /* if theta123 < 0.0 it is just SLIGHTLY */
              theta123=0.0;      /* < 0.0 and should should be corrected to 0 */

            if (theta123 > PI)
              theta123=PI;

            if (theta123 >= threshrad)
              {
              discardp2=TRUE;
              curvcrit++;
              (void) fprintf (stderr, "%s: curvcrit: theta123=%f\n", pname,theta
123);

              }
            else
              discardp2=FALSE;
            }

          if (discardp2)
            {
            p2lat=p3lat;                    /* discard p2 */
            p2long=p3long;
            }

          else
            {
            retainedlat[nretained]=p2lat;   /* retain p2 */
            retainedlong[nretained]=p2long;
            nretained++;
            p1lat=p2lat;
            p1long=p2long;
            p2lat=p3lat;
            p2long=p3long;
            }
          }

        /* Lastcheck: Check distance between last retained point and p3: */

        p2lat=retainedlat[nretained-1];
        p2long=retainedlong[nretained-1];

        d=dist(p2long,p2lat,p3long,p3lat);

        if (d >= maxthreshdist)
          {
          (void) fprintf (stderr, "%s: lastcheck: maxdistcrit violation\n",pname
);
          maxdistcrit++;
          }
```

```
        else
          {
          retainedlat[nretained]=p3lat;
          retainedlong[nretained]=p3long;
          nretained++;
          }

        if (nretained > 2)  /* Throw out sets with nretained=2 */
          {
          /* Write rank, nretained to output file:  */

          if (fprintf(ofp, "%d %d\n",rank,nretained) == EOF)
          {
          (void) fprintf (stderr, "%s: fprintf: could not write rank, nretained
to stdout\n",pname);
          return -1;
          }

          for (i=0;i<nretained;i++)
            {
            if (fprintf(ofp, "%f %f\n",retainedlat[i],retainedlong[i]) == EOF)
              {
              (void) fprintf (stderr, "%s: fprintf: could not write retainedlat[
%d], retainedlong[%d] to stdout\n",pname,i,i);
              return -1;
              }
            }
          }
        }

      totalretained=totalretained+nretained;

      (void) fprintf (stderr, "%s: nretained=%d, ",pname,nretained);
      (void) fprintf (stderr, "pctretained=%f\n",(float)nretained/(float)npts);
      (void) fprintf (stderr, "%s: curvcrit=%d, mindistcrit=%d, maxdistcrit=%d\n
\n",pname,curvcrit,mindistcrit,maxdistcrit);
      }
    }
    (void) fprintf (stderr, "%s: done\n",pname);
    (void) fprintf (stderr, "%s: totalpoints=%d, totalretained=%d, pcttotalretai
ned=%f\n",pname,totalpoints,totalretained,(float)totalretained/(float)totalpoint
s);
    (void) fclose (ifp);
    (void) fclose (ofp);
    return 0;
  }
```

Appendix C

bcor Program Source Code Listing

```
/* BCOR:  Corrects boundary crossing errors

   Written by Jim Kilgore
   Naval Research Laboratory  */


#include "bcor.h"

#define MAXPOSNS 20000

struct position p[MAXPOSNS];
extern struct _iobuf _iob[];

main(argc, argv)
int argc;
char **argv;
  {
  int fsize, eof = FALSE, rank, i, npts, cnt=0, fstat(), completed;
  long ftell(), fpos;
  FILE *ifp=stdin,*ofp=stdout,*fopen();
  char *pfname=argv[0];
  void bc();
  struct stat buf;

  if (argc != 1)
    {
    (void) fprintf (stderr, "usage: %s < ifname > ofname\n", pfname);
    return -1;
    }

  (void) fstat (fileno (ifp), &buf);

  fsize =  buf.st_size;

  while (!eof)
    {
    if (fscanf(ifp,"%d%d",&rank,&npts) == EOF)
      {
      eof=TRUE;
      }

    else
      {
/*
      fpos = ftell (ifp);
      completed = (int) ((float) fpos/ (float) fsize * 100.);
      (void) fprintf (stderr, "\rcompleted: %d pct", completed);
*/

      if (npts > MAXPOSNS)
        {
        (void) fprintf (stderr, "\n%s: %d > MAXPOSNS: MAXPOSNS too small\n",pfna
me, npts);
        return -1;
        }

      for (i=0;i<=npts-1;i++)
```

```
        {
        if (fscanf(ifp,"%f%f",&p[i].lat,&p[i].lon) == EOF)
          {
          (void) fprintf (stderr, "\n%s: unexpected EOF while reading lat, lon\n
",pfname);
          return -1;
          }
        }
      (void) bc (&cnt, npts, rank, &p[0], ofp);
      }
    }
/*
  (void) fprintf (stderr, "\rcorrections=%d     \n", cnt);
*/
  return 0;
  }
```

bc Program Source Code Listing

```
#include "bcor.h"


void bc (cnt, n, rank, pos, ofp)
int n, rank, *cnt;
struct position pos[];
FILE *ofp;
  {
  int i,j;
  register float d, dx1, dx2, dx, dy, dy1;
  struct position p0bound, p1bound;

  for (i=1; i<n; i++)
    {

    d = sqrt ((pos[i].lon-pos[i-1].lon)*(pos[i].lon-pos[i-1].lon)+(pos[i].lat-po
s[i-1].lat)*(pos[i].lat-pos[i-1].lat));

    if (d > MAXDIST)
      {

      /* p0bound is new tail of first segment, p1bound is head */
      /* of second segment */

      if (pos[i-1].lon > 0.)
        {
        p0bound.lon =  180.;
        p1bound.lon = -180.;
        }

      else
        {
        p0bound.lon = -180.;
        p1bound.lon =  180.;
        }


      /* Compute boundary point by simple interpolation: */

      dx1 = p0bound.lon - pos[i-1].lon;
      dx2 = p1bound.lon - pos[i].lon;
      dx  = dx1 - dx2;
      dy  = pos[i].lat - pos[i-1].lat;
```

```
      dy1 = dy/dx*dx1;
      p0bound.lat = p1bound.lat = dy1 + pos[i-1].lat;


      /* Write rank, n points in first segment: */

      (void) fprintf (ofp, "%d %d\n", rank, i+1);


      /* Write first segment to file, omitting tail point p0bound: */

      for (j=0; j<i; j++)
        (void) fprintf (ofp, "%f %f\n", pos[j].lat, pos[j].lon);


      /* Now write tail point p0bound to file: */

      (void) fprintf (ofp, "%f %f\n", p0bound.lat, p0bound.lon);


      /* Set pos[i-1] equal to head point p1bound of second segment, */
      /* destructively altering old value.  That's ok because we never */
      /* use old value again: */

      pos[i-1].lat = p1bound.lat;
      pos[i-1].lon = p1bound.lon;


      ++*cnt;   /* cnt is number of corrections made */


      /* Recursive call to bc starting with pos[i-1]: */

      (void) bc (cnt, n-i+1, rank, &pos[i-1], ofp);

      return;
      }
    }


  /* Do this if segment passed to bc doesn't need to be split: */

  (void) fprintf (ofp, "%d %d\n", rank, n);
  for (i=0; i<n; i++)
        (void) fprintf (ofp, "%f %f\n", pos[i].lat, pos[i].lon);
  return;
  }
```

bcor.h Listing

```
#include <stdio.h>
#include <math.h>
#include <sys/types.h>
#include <sys/stat.h>

#define TRUE 1
#define FALSE 0

#define MAXDIST 350.

struct position
       {
```

```
        float lat;
        float lon;
        };
```

Makefile Listing

```
CFLAGS = -fswitch
LFLAGS = -lm


main : bcor

bcor : bcor.o bc.o
        cc ${CFLAGS} bcor.o bc.o -o bcor ${LFLAGS}

bcor.o : bcor.c bcor.h
        cc ${CFLAGS} -c bcor.c

bc.o : bc.c bcor.h
        cc ${CFLAGS} -c bc.c
```

Appendix D

draw Program Source Code Listing

```
/* DRAW: a quick and dirty drawing program

   cc -fswitch draw.c -o dr -L./lib -lcore -lsunwindow -lpixrect -lm

   Note that lib and include directories are included in $PWD because
   SunCore is no longer supported as of OS 4.0 release and those files
   do not come with OS distribution.

   Written by Jim Kilgore
   Naval Research Laboratory */


#include "include/usercore.h"
#include <stdio.h>

#define BUFLEN 100000

int pixwindd(),bw2dd(),cg2dd();
struct vwsurf vwsurf = DEFAULT_VWSURF(bw2dd);

float lat[BUFLEN],lon[BUFLEN];

main(argc,argv)
int argc;
char **argv;
  {
  FILE *ifp,*fopen();
  float aspect_ratio,r[125],g[125],b[125],height,width;
  double latmin,lonmin,latmax,lonmax, atof();
  int eof,i,n,ir,ig,ib,rank,rankmax,npts,nfiles;
  char *file,*pname=argv[0];
  void cleanup();

  if (argc < 8)
    {
    (void) fprintf (stderr, "usage: %s m|M|c rankmax latmin lonmin latmax lonmax
 fname1 [fname2 fname3...]\n",pname);
    return -1;
    }

  nfiles=argc-7;

  if (*argv[1] == 'c')
    {
    vwsurf.dd=cg2dd;
    height=900;
    width=1152;
    }

  else if (*argv[1] == 'm')
    {
    vwsurf.dd=bw2dd;
    height=900;
    width=1152;
    }

  else if (*argv[1] == 'M')
    {
```

```
    vwsurf.dd=bw2dd;
    height=1280;
    width=1600;
    }

  else
    {
    (void) fprintf (stderr, "usage: %s b|B|c rankmax latmin lonmin latmax lonmax
fname1 [fname2 fname3...]\n",pname);
    return -1;
    }

  rankmax=atoi(argv[2]);

  latmin = atof (argv[3]);
  lonmin = atof (argv[4]);
  latmax = atof (argv[5]);
  lonmax = atof (argv[6]);

  /* Fill screen for quasi-equirectangular projection: */

  aspect_ratio=height/width;

  /* Gibberish for Core: */

  (void) initialize_core(BASIC,NOINPUT,TWOD);
  (void) initialize_view_surface(&vwsurf,FALSE);
  (void) select_view_surface(&vwsurf);
  (void) set_ndc_space_2(1.0,aspect_ratio);
  (void) set_viewport_2(0.0,1.0,0.0,aspect_ratio);
  (void) set_window (lonmin,lonmax,latmin,latmax);

  /* Define 125 colors: */

  n=0;
  for (ir=0;ir<=4;ir++)
    {
    for (ig=0;ig<=4;ig++)
      {
      for (ib=0;ib<=4;ib++)
        {
        r[n]=(float)ir/4.;
        g[n]=(float)ig/4.;
        b[n]=(float)ib/4.;
        n++;
        }
      }
    }

  (void) define_color_indices(&vwsurf,0,124,r,g,b);

  (void) set_line_index(124);
  (void) set_fill_index(124);
  (void) set_text_index(124);

  (void) create_temporary_segment();

  for (n=1;n<=nfiles;n++)
    {
    file=argv[6+n];

    if ((ifp=fopen(file,"r")) == NULL)
      {
      (void) fprintf (stderr, "%s: fopen: could not open \"%s\" for read\n",pnam
```

```
e,file);
      (void) cleanup();
      return -1;
      }
    (void) fprintf (stderr, "%s: drawing \"%s\"\n",pname,file);
    eof=FALSE;

    while (!eof)
      {
      if (fscanf(ifp,"%d%d",&rank,&npts) == EOF)
        eof=TRUE;
      else
        {
        if (npts > BUFLEN)
          {
          (void) fprintf (stderr, "%s:  sorry, npts=%d > BUFLEN=%d\n",pname,npts
,BUFLEN);
          (void) cleanup();
          return -1;
          }
        for (i=0;i<npts;i++)
          {
          if (fscanf (ifp,"%f%f",&lat[i],&lon[i]) == EOF)
            {
            (void) fprintf (stderr, "%s: unexpected EOF while reading lat, lon\n
",pname);
            (void) cleanup();
            return -1;
            }
          }
        if (rank <= rankmax)
          {
          (void) move_abs_2(lon[0],lat[0]);
          (void) polyline_abs_2(&lon[1],&lat[1],npts-1);
          }

        }
      }
      (void) fclose(ifp);
    }

  (void) fprintf (stderr,"Hit RETURN to clear screen and quit:\n");
  (void) getchar();
  (void) cleanup();
  return 0;
  }

void cleanup()
  {
  /* Reset colors */

  static float r[] = {1.0},
               g[] = {1.0},
               b[] = {1.0};
  (void) define_color_indices(&vwsurf,0,0,r,g,b);
  (void) close_temporary_segment();

  /* shut down core */

  (void) deselect_view_surface(&vwsurf);
  (void) terminate_core();
  }
```

Appendix E

suntokee Program Source Code Listing

```
/* SUNTOKEE:  converts a Sun monochrome (1 bit deep) standard
   rasterfile (rasterfile(5) format to a bitmap file in
   Xerox Interlisp-D format which is compatible with KEE.  Sun
   color rasterfiles (8 bits deep) are not accepted and must be
   first converted to a 1-bit deep rasterfile using
   rasfilter8to1(1).

   Below is an example of the syntax.  Bitmapname is the print
   name that will be given to the bitmap.

   suntokee bitmapname < rasterfname> bitmapfname

   Written by Jim Kilgore
   U. S. Naval Research Laboratory
   February, 1989

   Modified 8/9/90 to correctly read and convert rasterfiles whose
   ras_width is not an even multiple of 8. */



#include <rasterfile.h>
#include <stdio.h>
#include <sys/file.h>

main(argc, argv)
int argc;
char **argv;
  {
  FILE *ofp=stdout,*fopen();
  int lineno,pixelno,i,ifd=fileno(stdin);
  struct rasterfile rf;
  char buf,bufhi,buflo,*pname=argv[0],*bitmapname=argv[1];

  if (argc != 2)
    {
    (void) fprintf (stderr, "usage: < rasterfname %s bitmapname > bitmapfname\n"
, pname);
    return (-1);
    }

  /* Read the header and print it out:  */

  if (read (ifd,(char *) &rf,sizeof(rf)) == NULL)
    {
    (void) fprintf (stderr, "%s: read: unexpected EOF while reading header struc
ture\n", pname);
    return (-1);
    }

  /* Magic number must be correct:  */

  if (rf.ras_magic != RAS_MAGIC)
    {
    (void) fprintf (stderr, "%s: wrong magic number\n",pname);
    return (-1);
    }
```

```
  /* ras_length must be > 0:  */

  if (rf.ras_length <= 0)
    {
    (void) fprintf (stderr, "%s: bad header record, rf.ras_length must be > 0\n"
, pname);
    return (-1);
    }

  /* Rasterfile depth must be 1, i.e., monochrome:  */

  if (rf.ras_depth != 1)
    {
    (void) fprintf (stderr, "%s: sorry, rf.ras_depth must = 1\n",pname);
    return (-1);
    }

  /* Rasterfile must be type RT_STANDARD, i.e., not RT_BYTE_ENCODED,
     RT_FORMAT_RGB, etc.  */

  if (rf.ras_type != RT_STANDARD)
    {
    (void) fprintf (stderr, "%s: sorry, rasterfile must be standard format image
\n",pname);
    return (-1);
    }

  lineno=0;
  pixelno=0;

  (void) fprintf (ofp,"(in-package \'kee)\n");
  (void) fprintf (ofp,"(defparameter %s\n(expr-to-bitmap \'(%d %d 1\n\"",bitmapn
ame, rf.ras_width, rf.ras_height);


  for (i=1;i<=rf.ras_length;i++)
    {
    if (read(ifd,(char *) &buf,1) == NULL)
      {
      (void) fprintf (stderr, "\n%s: unexpected EOF while reading image\n",pname
);
      return (-1);
      }

    /* bufhi = upper 4 bits of buf; buflo = lower 4 bits of buf: */

    bufhi = (buf >> 4) & 0xf;
    buflo = buf & 0xf;

    /* Translate so bufhi and buflo lie within range of 0x40 and 0x4f
       (between ASCII @ and O):                                        */

    bufhi += 64;
    buflo += 64;

    (void) fprintf (ofp,"%c%c",bufhi,buflo);

    pixelno += 8;

    if (pixelno >= rf.ras_width)   /* >= because rf.ras_width may not be even mu
ltiple of 8    */
      {                            /* if so, screendump pads byte with zeros.
              */
      pixelno=0;
```

```
        lineno++;
        (void) fprintf (stderr, "\r%s: completed: %d pct", pname, (int) ((float) l
ineno / (float) rf.ras_height * 100.));
        (void) fprintf (ofp,"\"");

        if (lineno == rf.ras_height)
          (void) fprintf (ofp,")))\n");

        else
          (void) fprintf (ofp,"\n\"");
        }

      }

    (void) close(ifd);
    (void) fclose(ofp);

    (void) fprintf (stderr, "\n%s: done\n",pname);

    return 0;

    }
```

Appendix F

Listing of File *box2*

```
1 5
-90 -180
90 -180
90 180
-90 180
-90 -180
```

Listing of File *box3*

```
1 5
32 60
23 60
23 42
32 42
32 60
```

Listing of File *box4*

```
1 5
29.5 47
30.5 47
30.5 49
29.5 49
29.5 47
```